МОДЕЛИРОВАНИЕ РАСШИРЕННОЙ ЛЕММАТИЗАЦИИ ДЛЯ РУССКОГО ЯЗЫКА НА ОСНОВЕ МОРФОЛОГИЧЕСКОГО ПАРСЕРА TNT-RUSSIAN

Шаврина Т. О. (rybolos@gmail.com)[1],
Сорокин А. А. (alexey.sorokin@list.ru)[1,2,3]

[1]МГУ им. М. В. Ломоносова, Москва, Россия;
[2]МФТИ, Москва, Россия; [3]РГГУ, Москва, Россия

Автоматические методы морфологического анализа и лемматизации, предназначенные для литературного русского языка, могут давать невысокие результаты, будучи применёнными к так называемым социальным медиа (микроблоги, социальные сети и т. д.). Одной из причин является орфографическая вариативность текстов в социальных медиа, зачастую вызванная опечатками. Мы предлагаем интегрировать модуль исправления опечаток в алгоритм морфологического анализа на примере Генерального интернет-корпуса русского языка (ГИКРЯ), что позволит осуществить расширенную лемматизацию. Также в работе предлагается новый алгоритм исправления опечаток, основанный на взвешенном расстоянии Левенштейна и проводится анализ типичных нарушений орфографической нормы в текстах социальных медиа.

Ключевые слова: расширенная лемматизация, язык социальных медиа, исправление опечаток, расстояние Левенштейна

MODELING ADVANCED LEMMATIZATION FOR RUSSIAN LANGUAGE USING TNT-RUSSIAN MORPHOLOGICAL PARSER

Shavrina T. O. (rybolos@gmail.com)[1],
Sorokin A. A. (alexey.sorokin@list.ru)[1,2,3]

[1]Lomonosov Moscow State University, Moscow, Russia
[2]Moscow Institute of Science and Technology, Moscow, Russia
[3]Russian State University of Humanities, Moscow, Russia

Automatical morphological parsers and lemmatizers learnt on literature language sometimes show decent performance for social media texts (including microblogs, social network, etc.). Partially it is due to orthographical variability of social media, often caused by simple misprints. On the basis of General Internet corpora of Russian language (GICR) and TnT morphological parser we show how to integrate a spelling corrector into the pipeline of morphological parser which allows us to perform advanced lemmatization. We present an algorithm of spelling correction based on Levenshtein distance and give a preliminary classification of common orthographical deviations for Russian social media corpus.

Key words: advanced lemmatization, social media language, spelling correction, Levenshtein distance

Одним из основных источников новых текстов в Интернете являются так называемые «социальные медиа» (social media), к которым относятся, в частности, блоги, микроблоги и посты в соцсетях. Случайно взятый из социальных медиа текст с большой долей вероятности содержит ошибки как в орфографии, так и в пунктуации, а также ошибки капитализации. Это приводит к появлению дополнительной вариативности, при описании и обработке которой плохо работают методы, ориентированные на русский литературный язык.

На сегодняшний момент существует несколько крупных русскоязычных Интернет-корпусов — это корпус университета Лидс (http://corpus.leeds.ac.uk/internet.html), Sketch Engine (http://www.sketchengine.co.uk/documentation/wiki/Corpora/TenTen/ruTenTen) и пока недоступный для открытого доступа Генеральный интернет-корпус русского языка (ГИКРЯ, http://www.webcorpora.ru/). При обработке текста (в частности, лемматизации и морфологическом разборе) ни в одном из данных корпусов не проводится исправление опечаток, что, как следствие, снижает качество автоматической обработки текста и достоверность корпусной разметки. Хотя безусловная коррекция противоречит идее о фиксации языковой вариативности [Belikov et al., 2013], мы считаем, что опциональное использование коррекции могло бы оказаться весьма полезным. Хотя большой Интернет-корпус, как отмечено в [Norvig 2009], идеален для обучения автоматчиеского орфографического корректора, проблемам исправления опечаток в применении к Веб-корпусам посвящено незаслуженно мало работ ([Popescu, Phuoc An Vo 2014], [Norvig 2009]). В основном модели исправления опечаток разрабатывались для расширения поисковых запросов и автоматического извлечения информации ([Bajtin 2008], [Farooq, Grzegorz 2005], [Norvig 2010], [Panina, Baitin, Galinskaya 2013], [Segalovich 2013]), а также для типографских нужд ([Damerau, Mays 1989], [Peterson 1986]).

Нашей основной задачей является «расширенная лемматизация», состоящая в определении леммы для заведомо относящихся к ней слов, которые в то же время могут формально не являться её словоформами. Так, любой лингвист отнесёт к лемме «дневник» формы «дневника», «дневник», «Дневник», «дНЕВНИК», «днивник» и «денвник», хотя только первые две или три из них могут присутствовать в словаре. Отметим, что доля ошибок в корпусе ГИКРЯ

достаточно велика: так, около 7,8 % несловарных слов являются на самом деле результатом опечаток в словарных словах. При попытке их приведения к правильной лемме возникает несколько проблем:

1. Поскольку результаты исправления опечаток будут подаваться на вход морфологическому анализатору, словарь, используемый корректором, должен быть максимально близок к словарю анализатора. Словарь ГИКРЯ представляет собой объединения словаря mystem и автособираемого словаря tnt-russian. Поэтому применение внешних словарей и программ в данном случае затруднено.
2. Более того, сам словарь ГИКРЯ содержит опечатки.
3. Словарь ГИКРЯ весьма велик по объёму (7 000 000 слов), что предъявляет высокие требования к оптимизации алгоритмов поиска в словаре.

В данной работе мы предлагаем контекстно-независимую модель расширенной лемматизации для словоформ русского языка, основанную на морфологической разметке, предоставляемой парсером TnT-Russian [Sharoff, Nivre 2011]. Опечатками мы считаем отклонение от задуманной словоформы, отчетливо осознаваемое автором как отклонение — оно может быть как следствием ошибочного набора, так и следствием намеренного искажения (изменения регистра, замена на символы, цифры). Мы не учитываем в модели орфографические ошибки, так как для них существует большое количество свободно распространяемых программ(aspell, yspell и др.), однако при этом не проводим границы между различением намеренной и ненамеренной вариативности.

1. Из чего состоит русский текст

Чтобы понять, какими бывают опечатки, необходимо предварительно определить, из чего в широком понимании составляется русский текст. В текстах на русском языке встречаются последовательности из кириллических букв и пробелов («мама мыла раму»), из кириллических букв и дефиса («все-таки», «Тот-кого-нельзя-называть»), из кириллических букв и цифр («Т9», «Т-34»), из кириллических букв и символов («М@ша», «Иди сюда, ****»), символов и цифр («8 %», «§102»), букв латинского алфавита («pishu s telefona»), а также из массы прочих сочетаний вышеназванных элементов друг с другом, удаляющимся по шкале “русскоязычности» текста в сторону всё большей экзотичности. Оставим открытым вопрос, можно ли считать русским текстом, скажем, русские слова и целые фразы, случайно написанные в латинской раскладке. Технически вопрос обработки таких примеров уже решён с помощью программ переключения раскладки (например, Punto Switcher), а следует ли данную технологию включать в алгоритм расширенной лемматизации — зависит от состава и задач каждого конкретного корпуса. Заметим лишь, что мы однозначно не считаем русским текстом текст другого языка, написанный кириллицей, но считаем русским русскоязычный текст, написанный в латинской раскладке. Итак, главный наш критерий — это намерение автора написать текст по-русски.

Средства, которыми может быть написан русский текст, могут быть разными, но мы будем рассматривать такой состав его элементов:

1) кириллица [абвгдеёжзийклмнопрстуфхцчшщъыьэюя
АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ]
2) латиница [abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ]
3) дефис и апостроф (‘-)
4) знаки пунктуации [-.()?:”«»„”!;,…— –]
5) специальные символы (далее просто «символы») [‘\//|&=+*°¶§<>[]
{}_№%#~^@£$€¦]
6) пробел
7) цифры [0123456789]

Этот набор не покрывает все возможные варианты символов, встречающихся в интернет-текстах, так как существуют ещё и специальные символы юникода, слова в раскладке других языков (например, в качестве цитат) и т. д., но наш набор сводится к тому множеству элементов, которое можно получить на стандартной 105-клавишной клавиатуре в русской и латинской раскладке: (рис. 1)

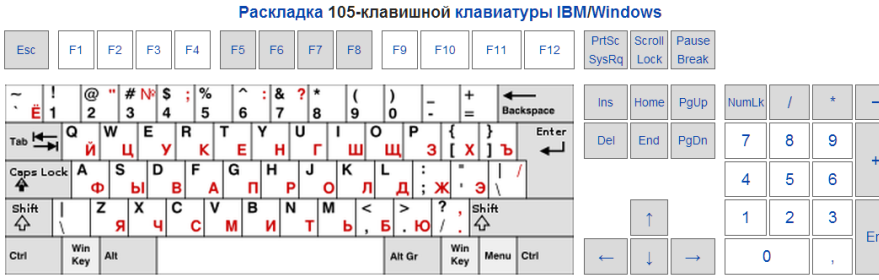

Рисунок 1

2. Специфика программной реализации

Разрабатываемую программу исправления опечаток предполагается использовать как один из элементов уже существующего алгоритма обработки текста в Генеральном интернет-корпусе русского языка на уровне лемматизация-морфология. В связи с этим опишем схему морфологического анализа на данном уровне. Морфологическая разметка осуществляется с помощью программы «TnT Russian», получающей на вход последовательность токенов и приписывающей им лемму, а также метку морфологической категории. При этом используется система морфологических меток, применявшаяся в проекте MULTEXT-East [Dimitrova et al., 1998]. При этом возможны два случая: поступившая на вход лемма присутствует в словаре («словарный») или её там нет.

Случай 1: *словарный*

Лемма и морфологическая метка определяются с помощью двух словарей, первый из которых сопоставляет словоформам метки морфологических категорий и их вероятности, а второй отображает данные метки в леммы. В первом словаре используется программа TreeTagger [Schmid 1994], обученная на подкорпусе НКРЯ со снятой омонимией; второй словарь также автоматически получен на подкорпусе НКРЯ со снятой омонимией, а затем дополнен при помощи словаря mystem [Segalovich 2003]).

Случай 2: *несловарный*

Если поступившей на вход словоформы нет в словаре, метки и лемма для неё определяются согласно словарю суффиксов и n-граммной модели. В таком случае слову дополнительно приписывается метка <guessed>. Таких случаев в среднем по корпусу примерно 2%.

Задача этого исследования состоит в создании контекстно-независимой модельи, приписывающую вариативные формы слова существующей единице словаря, отражающего стандартный русский язык. Модель встраивается в обработку текста программой TnT-Russian на этапе после морфологического анализа и сравнения со словарём словоформ, но перед передачей несловарной словоформы в модуль лемматизации по суффиксам. На рис. 2 показан предлагаемый алгоритм работы программы; рассматриваемый в данной работе этап расширенной лемматизации (РЛ) выделен красным.

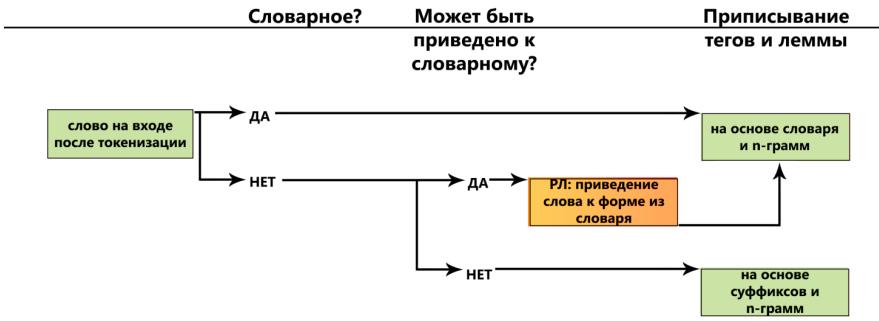

Рисунок 2. Схема работы TnT Russian с учетом расширенной лемматизации

3. Подготовка модели и словаря

3.1. Типы возможных ошибок

Мы провели предварительное исследование возможных опечаток на корпусе. Для исследования был выбран подкорпус ГИКРЯ, состоящий из текстов Живого Журнала (ЖЖ, m.livejournal.com) и имеющий объём ~10 млрд слов. Для 71 млн слов морфологический анализатор TnT-Russian не смог подобрать лемму из словаря. На случайной подвыборке из данного корпуса мы подробно проанализировали распределение несловарных слов, результаты анализа отражены в таблице 1.

Таблица 1. Распределение несловарных слов по типам (на материале ЖЖ, ГИКРЯ)

Тип несловарности	**%**
Ошибки анализатора ГИКРЯ (прежде всего причастия)	61,30
Несловарные слова русского языка (неологизмы сленг)	21,31
Имена собственные	7,85
Опечатки +намеренное отклонение от нормы (регистр, замена на символы,)	7,73
Ненамеренные орфографические ошибки	1,81

Если исключить из вышеприведённой таблицы ошибки анализатора, вызванные изменением обработки причастий в ГИКРЯ, то среди действительно несловарных словоформ получаем распределение 2. — 55 %, 3. — 20,2 %, 4. — 19,9 %, 5. — 4,69 %,что соотносимо с данными, полученными в работе [Farooq, Grzegorz 2005]). В нашей модели мы хотим исправлять прежде всего ошибки типа 4., не хотим трогать типы 1–3, и не интересуемся ошибки типа 5., признавая, что некоторые из них обрабатываются вместе с ошибками типа 4.

3.2. Проблемы с автоматически собранным словарём

Поскольку словарь TnT-Russian собирался автоматически, он содержит многие частотные опечатки. Необходима либо дополнительная очистка словаря, которая снизит его полноту, либо переход к контекстно-зависимой модели исправления опечаток, дополнительно учитывающей частоту словарных слов. Поскольку второй подход более сложен в реализации, мы предпочли снизить полноту словаря, оставив в нём только слова, распознаваемые анализатором mystem. Это привело к существенному уменьшению объёма словаря (до 250 000 слов) и, как следствие, ускорению работы корректора и повышению его точности. Это было сделано с целью тестирования самого алгоритма исправления опечаток.

В дальнейших исследованиях мы планируем усложнить модель для обработки несловарных слов, прежде всего привлекая контекстные и частотные соображения.

4. Алгоритм исправления опечаток

Для исправления опечаток используется вероятностная модель, основанная на взвешенном расстоянии Левенштейна [Damerau, 1964]; [Levenstejn, 1965]. Веса отдельных операций определяются с помощью EM-алгоритма [Dempster, 1979]. Данный подход является весьма распространённым в литературе по исправлению опечаток (см., например, [Kernighan, Church, Gale, 1990]). Для отбора наилучшего кандидата c из множества возможных кандидатов C используется байесовская модель канала связи:

$$p(c|w) = \frac{p(w|c)p(c)}{p(w)}$$

$$c = \underset{c \in C}{\operatorname{argmax}}\, p(c|w) = \underset{c \in C}{\operatorname{argmax}}\, p(w|c)p(c)$$

Таким образом, для полного описания модели необходимо задать, во-первых, алгоритм определения кандидатов, во-вторых, модель опечаток $p(c|w)$, в-третьих, словарную модель вероятности $p(c)$. Чтобы адекватно учесть априорную вероятность слов-кандидатов $p(c)$, необходимо перейти к логлинейной модели

$$\mathrm{c} = \underset{c \in C}{\operatorname{argmax}} \log p(w|c) + \lambda \log p(c),$$

где параметр λ дополнительно подбирается путём оптимизации на тестовой выборке. Однако это приводит к дополнительному усложнению модели, поэтому мы решили вообще не учитывать априорную вероятность кандидатов. Стоит отметить, что при дальнейшем улучшении модели величина $p(c)$ может стать контекстно-зависимой, например, в неё можно интегрировать n-граммную модель для морфологических меток.

4.1. Поиск кандидатов

В качестве слов-кандидатов рассматриваются все словарные слова на расстоянии 2 или меньше от данного. Как показал анализ случайной подвыборки опечаток, в 98% случаев правильный вариант действительно находится на таком расстоянии. При этом в случае, если существуют словарные слова на расстоянии 1, слова на расстоянии 2 в качестве кандидатов не рассматриваются. При вычислении расстояния ρ между словами элементарными операциями стоимости 1 считаются замены, удаления, вставки и перестановки соседних символов. При этом ни один символ не может дважды подвергаться этим операциям (в частности, ρ(AB, BCA) =3, а не 2). Данная мера близости (так называемое расстояние оптимального выравнивания, optimal alignment distance) не является метрикой, но может быть легко вычислена с помощью стандартного динамического алгоритма, что объясняет её использование в задачах исправления опечаток (например, [Brill, Moore, 2000]).

Отдельный интерес (с учётом большого объёма словаря) представляет поиск близких по расстоянию Левенштейна слов. Мы храним словарь в виде ациклического конечного автомата, полученного минимизацией префиксного бора, а для поиска используем стандартный алгоритм Офлазера из работы [Oflazer, 1995]. Для упрощения поиска используется A*-поиск с h_2-эвристикой из работы [Hulden, 2009]: в каждом состоянии ациклического автомата хранятся символы, встречающиеся на глубине не более 2 от данного состояния (например, если автомат распознаёт слова actor, actress и across, то в состоянии, соответствующем префиксу ac, хранятся символы t, r, o и e). Отметим, что использование конечного автомата позволяет и обрабатывать опечатки вида 2 слова → 1 слово (например, поло вина → половина), для этого автомат нужно дополнить рёбрами из всех конечных состояний в начальное, помеченными пробельными символами.

4.2. Вероятностная модель поиска оптимального кандидата

На данном этапе производится поиск оптимального словарного кандидата *c*, максимизирующего величину $p(w|c)$. Эту вероятность можно переписать в виде

$$p(w|c) = \frac{1}{Z} \sum_{n, w_i, c_i} p(w_1|c_1) \dots p(w_n|c_n)$$

Здесь Z — нормировочный множитель, w_1, …, w_n, c_1, …, c_n — это сегменты оптимального выравнивания между c и w: например, для пары *c*=мак, *w*=матк мы имеем $c_1 = w_1 = м$, $c_2 = w_2 = а$, $c_3 = \varepsilon$, $w_3 = т$, $c_4 = w_4 = к$. В выравнивании мы допускаем пары вида $a \leftrightarrow b$ (замены), $a \leftrightarrow \varepsilon$ (удаления), $\varepsilon \leftrightarrow a$ (вставки), а также $ab \leftrightarrow ba$ (перестановки). Отметим, что поиск оптимального выравнивания между *w* и *c* можно осуществлять с помощью алгоритма, вычисляющего расстояние Левенштейна, если взять в качестве штрафа за замену сегмента c_i на w_i величину $-\log p(w_i|c_i)$. Однако в формуле суммирования нас интересуют не только оптимальные выравнивания, но и все выравнивания с достаточно большой вероятностью. Поскольку за счёт наличия вставок и удалений число выравниваний растёт экспоненциально от длины слова, мы вначале оцениваем максимальную стоимость выравнивания, находя для каждого слова-кандидата *c'* оптимальное выравнивание между *w* и *c'*, после чего берём максимальную стоимость R такого выравнивания по всем кандидатам *c'*. После этого суммируются лишь выравнивания, чья стоимость не превышает *R*. Такие выравнивания можно находить обычным жадным поиском в глубину, отсекая частичные выравнивания, чья стоимость стала больше, чем *R*.

Алгоритм исправления опечаток позволяет не только вернуть наиболее вероятное слово-кандидат, но и вероятности других кандидатов. В случае, если ни один кандидат не имеет вероятности, превышающей $\frac{2}{|C|}$, где через *C* обозначено множество кандидатов, мы считаем, что опечатки не произошло, а на вход поступило корректное несловарное слово (например, неологизм).

4.3. Обучение модели

Для применения вероятностной модели необходимо оценить вероятности $p(w_i|c_i)$. Мы будем определять их автоматически на основе EM-алгоритма [Kernighan, Church, Gale, 1990]. Мы представляем данную вероятность в виде $p(\tau)\ p_\tau(w_i|c_i)$, где τ — это тип операции (вставка, замена, удаление или перестановка), а $p_\tau(w_i|c_i)$ — соответствующая условная вероятность, берущаяся только по множеству операций данного типа. Достоинство EM-алгоритма состоит в том, что он работает по методу обучения без учителя (то есть не требует наличия корпуса с исправленными опечатками).

На вход EM-алгоритму подаётся корпус слов с опечатками, содержащий слова $v_1, \ldots, v_N$ (в нашем случае N = 56 000). Для каждого слова v_j находятся кандидаты $u_{j1}, \ldots, u_{jm_j}$ и их вероятности $p_{j1}, \ldots, p_{jm_j}$ в соответствии с алгоритмом из раздела **4.2**, а также находятся соответствующие оптимальные выравнивания. После этого для каждого типа операции τ подсчитывается, сколько раз она встречалась в выравниваниях, при этом если кандидат u для слова v имеет вероятность p, то все пары, входящие в это выравнивание, считаются с весом p. Таким образом, для каждой операции τ мы получаем соответствующий счётчик N_τ. После этого вероятность данной операции вычисляется по формуле $p(\tau) = \frac{N_\tau}{\sum N_\tau}$. Аналогичным образом пересчитываются и сами вероятности $p_\tau(w_i|c_i)$, например,

$$p_{1\leftrightarrow 1}(a|\text{б}) = \frac{N(a|\text{б})}{\sum N(\cdot|\text{б})},$$

где суммирование ведётся по всем символам алфавита.

Вероятности инициализируются случайным образом, при этом значениями, близкими к 1, инициализируется вероятность замены, а для каждого символа a — вероятность замены $a \to a$. На тестовой выборке алгоритм сходился достаточно быстро, в частности, уже после 15 проходов корпуса значения вероятностей переставали меняться.

4.4. Результаты тестирования

Ниже приводится оценка полноты работы существующих систем на тестовом корпусе из опечаток ГИКРЯ (тест содержал заведомо только слова с опечатками, поэтому точность оценивать нерелевантно). За правильный ответ системе присваивался +1 балл, за ошибку — 0 баллов. Для систем, где возможно увидеть несколько вариантов исправления (yspell, aspell, Word, GICR lemming), начислялось 0,5 баллов, если правильный ответ был среди второго/третьего кандидатов.

Программа	yspell	aspell	google	MS Word 2007	GICR lemming*	Яндекс
Полнота на тестовом корпусе опечаток	0,862	0,694	0,765	0,709	0,748	0,829

Также мы приводим оценку качества для опечаток вида 1 слово → 1 слово.

Программа	yspell	aspell	google	MS Word 2007	GICR lemming*	Яндекс
Полнота на тестовом корпусе опечаток	0,876	0,701	0,801	0,801	0,795	0,842

4.5. Применение корректора к корпусу

В результате применения программы-корректора было получено следующее распределение типов ошибок (сегмент — Живой журнал, ГИКРЯ):

Таблица 2. Типы ошибок в корпусе ЖЖ ГИКРЯ

Тип ошибки	%
Пропущенная буква	41,79100
Неверная буква	18,80600
Лишняя буква	18,20900
Отсутствие пробела	11,34330
Перестановка букв	7,76119
Отсутствие капитализации	1,49254
Все остальные типы	0,59701

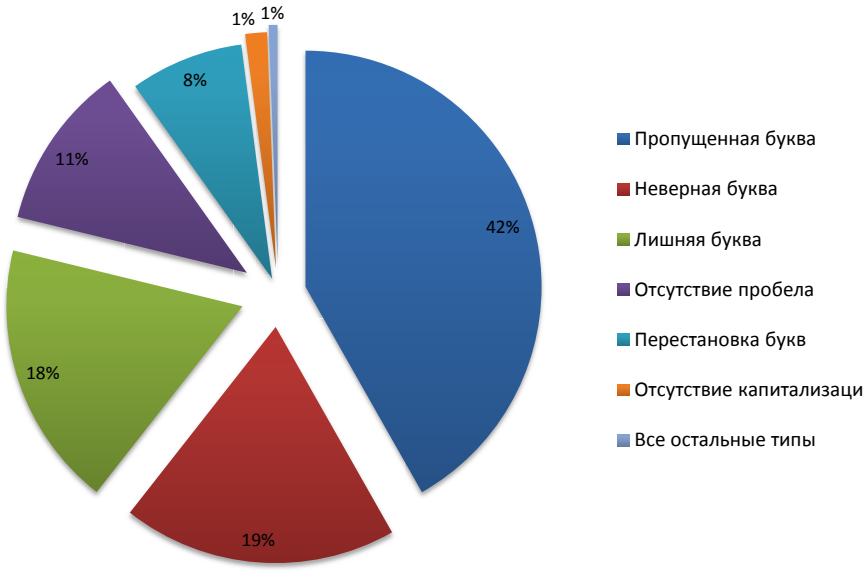

Рисунок 3. Распределение типов ошибок на сегменте ЖЖ ГИКРЯ

Мы ожидаем повышения качества разметки в среднем на 0,2-0,5% (что даёт от 40 до 100 млн токенов на всём корпусе). Несмотря на то, что доля исправлений относительно невелика, даже столь незначительное улучшение может существенным образом повлиять на качество лемматизации и морфологического анализа. Поскольку качество коррекции пока далеко от стопроцентного (и более того, стопроцентное качество в принципе не может быть достигнуто), то модуль исправления опечаток может быть только опциональным, т. е. у пользователя будет возможность его включения/отключения в зависимости от нужд исследования.

5. Заключение

Нами построена модель исправления опечаток в автоматически собираемом корпусе, дающая результаты, сравнимые с доступными аналогами. Это открывает новые возможности для исследования языковой вариативности. Так, после применения корректора в корпусе остаётся специальная метка, что позволяет исследователю изучать как отклонения от языковой нормы при написании той или иной словоформы, так и эволюцию самой орфографической нормы.

Наша модель допускает дальнейшие улучшения. Например, основной причиной проигрыша аналогичным программам является намеренный отказ от использования контекста. Интеграция контекстно-зависимой модели позволила бы улучшить точность модели, особенно в случаях ошибок типа 2 слова $\rightarrow$ 1 слово. Также перспективным выглядит использование частичного обучения в EM-алгоритме, что позволило бы как ускорить вычисления, так и улучшить качество модели.

Помимо изучения языковой вариативности, наш корректор может быть полезен и в решении других вопросов. Например, его можно использовать для уточнения автоматически собранного словаря TnT-Russian, а также для решения других задач.

Литература

1. *Bajtin A.* (2008), Search query correction in Yandex [Ispravlenie poiskovykh zaprosov v Yandekse], Russian Internet technologies [Rossijskie Internet-tekhnologii], 2008.
2. *Belikov V., Kopylov N., Piperski A., Selegey V., Sharoff S.* (2013), Corpus as language: from scalability to register variation [Korpus kak yazyk: ot masshtabiruemosti k differentsial'noj polnote], Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialog 2013" [Komp'yuternaya Lingvistika i Intellektual'nye Tekhnologii: Trudy Mezhdunarodnoy Konferentsii "Dialog 2013"], Bekasovo, pp.84–96
3. *Belikov V., Kopylov N., Piperski A., Selegey V., Sharoff S.,* (2013), Big and diverse is beautiful: A large corpus of Russian to study linguistic variation. Proceedings of Web as Corpus Workshop (WAC-8), Lancaster.

4. *Brill E., Moore Robert C.* (2000). An Improved Error Model for Noisy Channel Spelling Correction. Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, Association for Computational Linguistics, 2000, pp. 286–293.
5. *Damerau F. J.* (1964) A technique for computer detection and correction of spelling errors. Communications of the Association for Computing Machinery, Vol. 7, No. 3, pp. 171–176.
6. *Damerau F. J., Mays E.* (1989) An examination of undetected typing errors. Information Processing and Management, Vol. 25, No. 6, pp. 659–664.
7. *Dempster A. P., Laird N. M., Rubin D. B.* (1977) Maximum likelihood from incomplete data via the EM algorithm, Journal of the royal statistical society, Vol. 39, No. 1, pp. 1–38.
8. *Dimitrova, L., T. Erjavec, N. Ide, H.-J. Kaalep, V. Petkevič, and D. Tufiş* (1998) Multext-East: Parallel and Comparable Corpora and Lexicons for Six Central and Eastern European Languages. Proceedings of COLING-ACL '98. Montreal, Quebec, Canada.
9. *Farooq A., Grzegorz K.* (2005), Learning a Spelling Error Model from Search Query Logs. In Proc. of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), Vancouver, pp. 955–962.
10. *Hulden M.* (2009), Fast approximate string matching with finite automata, Natural language processing [Procesamiento del lenguaje natural], Vol. 43, pp. 57–64.
11. *Kernighan M. D., Church K. W., Gale W. A.* (1990) A spelling correction program based on a noisy channel model, Proceedings of the 13th conference on Computational linguistics, Volume 2, pp. 205-210.
12. *Kukich K.* (1992), Techniques for Automatically Correcting Words in Text. ACM Computing Surveys, Vol. 24, No. 4, pp. 377–439.
13. *Levenshtein V. A.* (1965) Binary codes capable of correcting deletions, insertions and reversals [Dvoichnye kody s ispravleniem udalenij, vstavok i zamen simvolov], Doklady of the Soviet Academy of Sciences [Doklady Akademij Nauk SSSR], 1965, Vol. 163, No. 4, pp. 845–848.
14. *Norvig P.* (2009) Natural Language Corpus Data, in Beautiful Data, O'Reilly Media, Chapter 14, pp. 219–243.
15. *Norvig P.* (2010) How to write a spelling corrector. Available at: http://norvig.com/spell-correct.html.
16. *Oflazer K.* (1996) Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction, Computational Linguistics, Vol. 22, No. 1, pp. 73–89.
17. *Panina M. F., Baitin A. V., Galinskaya I. E.* (2013) Context-independent autocorrection of query spelling errors. [Avtomaticheskoe ispravlenie opechatok v poiskovykh zaprosakh bez ucheta konteksta], Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialog 2013" [Komp'yuternaya Lingvistika i Intellektual'nye Tekhnologii: Trudy Mezhdunarodnoy Konferentsii "Dialog 2013"], Bekasovo, pp. 556–568.
18. *Peterson, J. L.* (1986) A note on undetected typing errors. Communications of the. ACM, Vol. 29, No. 7, pp. 633-637.

19. *Popescu O., Phuoc An Vo N.* (2014) Fast and Accurate Misspelling Correction in Large Corpora. Proceedings of EMNLP 2014: Conference on Empirical Methods in Natural Language Processing. Doha, Qatar.
20. *Schmid H.* (1994) Probabilistic Part-of-Speech Tagging Using Decision Trees. Proceedings of International Conference on New Methods in Language Processing, Manchester, UK.
21. *Segalovich I.* (2003) A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine, Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications (MLMTA), Las Vegas, pp. 273–280.
22. *Sharoff S., Nivre J.* (2011), The proper place of men and machines in language technology: Processing Russian without any linguistic knowledge. Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialog 2011" [Komp'yuternaya Lingvistika i Intellektual'nye Tekhnologii: Trudy Mezhdunarodnoy Konferentsii "Dialog 2011"], Bekasovo, pp. 591–605.

Приложение. Анализ типов возможных опечаток в подкорпусе ЖЖ ГИКРЯ

Типы рассматриваемой вариативности

В основном наша работа посвящена описанию алгоритма исправления опечаток и его применения к корпусу опечаток ЖЖ. Вместе с тем интересно изучить то, какие опечатки вообще могут встретиться в данном корпусе. Они могут быть самой разной природы; мы возьмём в рассмотрение те, которые отвечают двум критериям: 1) отклонение от нормы не связано с уровнем владения русским языком (так отсекаются орфографические ошибки) 2) это отклонение можно воспроизвести при помощи стандартной 105-клавишной клавиатуры (отсекаются экзотические случаи типа «ØĶØŁØ ΗΣЪΔ — ØЪΔ ΗΔÐ ΖΣΜŁÊŅ»).

Ниже приведён список достаточно частотных отклонений — некоторые из этих явлений встречаются в текстах всех жанров (например, опечатки в кириллице), некоторые характерны только для social media (ТеКсТ лЕсЕнКоЙ).

Отклонения в кириллице:

1. Замена (здесь и далее типы вариативности иллюстрируются примерами из корпуса и их текущей неидеальной обработкой морфологическим анализатором, также приводится желаемый результат лемматизации)

вхождение	метка	лемма	метка	желаемый результат
кододец	*Ncmsny*	*кододец*	*<guessed>*	*Колодец*

2. Пропуск

вхождение	метка	лемма	метка	желаемый результат
кзалось	*Vmis-snm-e*	*кзаться*	*<guessed>*	*Казаться*

3. Добавление

вхождение	метка	лемма	метка	желаемый результат
былаа	*Vmis-sfa-p*	*былаа*	*<guessed>*	*Быть*

4. Перестановка

вхождение	метка	лемма	метка	желаемый результат
товей	*Ncmsan*	*товей*	*<guessed>*	*Твой*

[Этап 1.0, предварительная очистка текста]
Отклонения, смешанные с латиницей:

5. Транслит

вхождение	метка	лемма	метка	желаемый результат
pishu	-	*pishu*		*Писать*

6. Замена на латинские буквы

вхождение	метка	лемма	метка	желаемый результат
profессонал	-	*profессонал*		*Профессионал*

7. Замена на латиницу как следствие ошибочной раскладки в начале набора слова.

вхождение	метка	лемма	метка	желаемый результат
gривет	-	*gривет*		*Привет*

Отклонения регистра:

8. Отсутствие капитализации
9. Ошибочная капитализация

вхождение	метка	лемма	метка	желаемый результат
МашкА	*Ncfsnn*	*машка*	*<guessed>*	*Машка (без <guessed>)*

10. Обратная капитализация — нивелирование последствий caps lock'a.

вхождение	метка	лемма	метка	желаемый результат
сАДИЛСЯ	*Vmis-smm-p*	*садиться*	*<guessed>*	*садиться (без <guessed>)*

11. Текст «лесенкой» и частичная капитализация

вхождение	метка	лемма	метка	желаемый результат
ДнЕвНиК	*Afpmsns*	*дневникий*	*<guessed>*	*дневник*

Отклонения с пробелом:

12. Лишний пробел: если два соседних слова несловарные, между ними убирается пробел и полученное слово проверяется по словарю.

вхождение	метка	лемма	метка	желаемый результат
девяти	*Mc--g*	*девять*	*<guessed>*	*Девятиэтажка*
этажка	*Ncfsnn*	*этажка*	*<guessed>*	*Девятиэтажка*

13. Отсутствие пробела: обрабатывается стандартной моделью исправления опечаток.

Отклонения с символами, числами:

14. Замена на символы, сходные визуально

вхождение	метка	лемма	метка	желаемый результат
}|{еня	*Npfsny*	*}|{еня*	*<guessed>*	*Женя*

15. Замена на символы или пунктуаторы, близкие на расположению на клавиатуре

вхождение	метка	лемма	метка	желаемый результат
Иванэ	*Npfsny*	*Иванэ*	*<guessed>*	*Иван ,*

16. Замена букв на числа

вхождение	метка	лемма	метка	желаемый результат
дево4ка	*Afpfsns*	*дево4ка*	*<guessed>*	*Девочка*

17. Лишний символ или знак пунктуации

вхождение	метка	лемма	метка	желаемый результат
*исследованиями**	*Ncmsan*	*исследованиями**	*<guessed>*	*исследование*